Universidad Zaragoza
1542

Trabajo de Fin de Máster en Ingeniería Industrial

SLAM visual robusto en entornos dinámicos combinando deep learning y consistencia multivista

Autor

Irene Ballester Campos

Directores

Javier Civera Sancho

Alejandro Fontán Villacampa

Klaus H. Strobl

Escuela de Ingeniería y Arquitectura

2020

SLAM visual robusto en entornos dinámicos combinando deep learning y consistencia multivista

RESUMEN

Los sistemas SLAM (acrónimo del inglés, *Simultaneous Localization and Mapping*), fundamentales para aplicaciones como la navegación autónoma, tienen como objetivo estimar el movimiento a partir de las imágenes tomadas por una cámara a la vez que construyen un mapa del entorno. El funcionamiento de estos sistemas supone que operan en un ambiente estático, por lo que la existencia de objetos en movimiento en la escena introduce importantes errores e inconsistencias en la construcción del mapa y la estimación de la trayectoria de la cámara.

Con el objetivo de mejorar la robustez y precisión de los sistemas de SLAM en entornos con elementos dinámicos, hemos desarrollado DOT (del inglés, *Dynamic Object Tracking*), un algoritmo de *front-end* integrable con sistemas de SLAM ya existentes. DOT combina una red neuronal y la geometría multivista para generar máscaras que identifican de forma robusta las partes dinámicas de cada imagen, de manera que el sistema SLAM no las emplee en su algoritmo de estimación. Para determinar qué objetos están en movimiento, en primer lugar se obtienen las máscaras de segmentación de los objetos potencialmente móviles mediante una red neuronal y, teniendo en cuenta el movimiento de la cámara, se realiza el *tracking* de los objetos segmentados mediante la minimización del error fotométrico.

Se ha evaluado DOT en combinación con ORB-SLAM 2 [1] en tres datasets públicos diseñados para la investigación en conducción autónoma. Los resultados muestran que la estrategia de DOT para estimar el movimiento de los objetos y generar las máscaras que identifican los objetos en movimiento permite mejorar la robustez y precisión un sistema de SLAM tanto en escenas dinámicas como estáticas.

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe entregarse en la Secretaría de la EINA, dentro del plazo de depósito del TFG/TFM para su evaluación).

D./Dª. Irene Ballester Campos ,en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Máster (Título del Trabajo)

"SLAM visual robusto en entornos dinámicos combinando deep learning y consistencia multivista"

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 24 de junio de 2020

Fdo: Irene Ballester Campos

Agradecimientos

A través de estas líneas quiero expresar mi más sincero agradecimiento a mis tutores Klaus y Javier, en primer lugar, por haberme dado la oportunidad de pasar estos meses en el Centro Aeroespacial Alemán rodeada de los mejores. Klaus, muchas gracias por hacerme sentir como en casa y por tu excelente labor de guía tanto en cuestiones técnicas como de *deustche Kultur*; y, Javier, estoy tremendamente agradecida por tus orientaciones y disponibilidad a lo largo de todo este tiempo.

El apoyo y asesoramiento de Alejandro han sido, sin duda, esenciales durante estos meses, así que quería darte las gracias por tu impagable ayuda en el desarrollo del proyecto (¡y en la comprensión de la intrincada lengua de Goethe!).

Por último, agradecer a mis compañeros de sala, Laura y Bastien, el excelente soporte informático recibido y, sobre todo, los buenos momentos dentro y fuera del DLR.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción y objetivos

1.1. Introducción

La auto-localización y construcción de mapas es una capacidad fundamental para la navegación autónoma de plataformas robóticas. Dicha capacidad es conocida comúnmente con el acrónimo SLAM, del término inglés *Simultaneous Localization and Mapping*. Para que la localización y el mapa sean útiles en la navegación, es habitual considerar que deberían estimarse en tiempo real, de manera incremental, y a partir únicamente de los sensores instalados en el robot. Una de las configuraciones con mayor potencial es el caso del SLAM visual para el que los sensores son principal, o exclusivamente, cámaras. Algunos ejemplos de aplicaciones de SLAM visual se muestran en la figura 1.1.

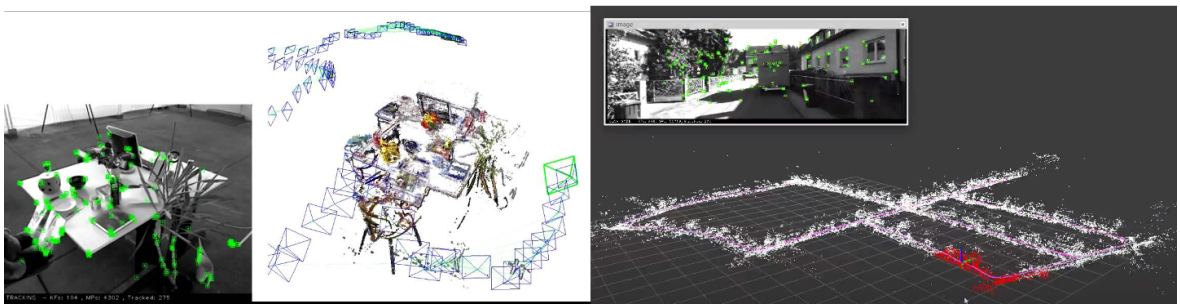

Figura 1.1: Aplicaciones de un sistema SLAM: reconstrucción del mapa 3D de la escena a partir de *features* (en verde en los fotogramas) y estimación de la trayectoria de la cámara. Izq.: Ejemplo de reconstrucción en espacios interiores. Dcha.: Ejemplo de reconstrucción para navegación de vehículos. Fuente: ORB-SLAM2 [2].

En la actualidad, la mayor parte de los sistemas de SLAM visual suponen un entorno estático, donde la posición relativa entre los puntos 3D de la escena no cambia, y el único movimiento corresponde al de la cámara. Con esta suposición, los algoritmos que estiman la posición y orientación de la cámara (en inglés, *pose*), atribuyen

el movimiento entre dos imágenes exclusivamente a la transformación relativa entre ambas. No contemplan, por tanto, la existencia de objetos en movimiento. En el mejor de los casos, algunos algoritmos los tratan como datos espurios, siendo ignorados tanto en la estimación del movimiento de la cámara como en el proceso de estimación del mapa. No obstante, como se muestra en la figura 1.2, esto no evita que en el tiempo que transcurre hasta su detección como objetos móviles, la información asociada se integre en la estimación suponiendo que es una escena rígida y se introduzcan errores e incoherencias en la estimación de la localización y el mapa.

Figura 1.2: Ejemplos de inconsistencias en el mapa cuando hay objetos dinámicos en la escena, mientras que las partes estáticas se reconstruyen correctamente, se generan inconsistencias en la zona donde la silla está girando o las personas se están moviendo. Fuentes: Izq.: MaskFusion [3]. Dcha.: Navvis [4]

.

Es más, en aquellos algoritmos de SLAM visual que basan la estimación del movimiento de la cámara (*pose tracking*) en el emparejamiento de un reducido número de puntos o características interesantes de la escena (*key-points o features*), los errores introducidos por elementos dinámicos pueden ser catastróficos e incluso producir el fallo del sistema.

El mundo de las aplicaciones reales en las que un robot debe desenvolverse está, en general, lejos de ser completamente estático: la navegación autónoma de vehículos como automóviles o drones, aplicaciones de realidad aumentada o tareas de exploración terrestre e incluso planetaria (donde la falta de características identificables en las imágenes vuelven precarios los sistemas de SLAM frente a sombras u otros robots). Todo ello plantea la necesidad de desarrollar un sistema de SLAM visual capaz de operar de forma robusta con elementos dinámicos.

1.2. Objetivos

El propósito de este trabajo es desarrollar un algoritmo de pre-procesamiento (*front -end*) que mejore la robustez de un sistema de SLAM visual en entornos en los que existen objetos en movimiento. El resultado (*output*) de este pre-procesamiento será una máscara que codifique las partes estáticas y dinámicas de cada imagen para que

el sistema no utilice las correspondencias que se encuentran en las regiones dinámicas. El trabajo incluye una validación experimental diseñada específicamente para evaluar la capacidad del sistema para reducir de forma efectiva los errores asociados a la estimación de movimiento y mapa del SLAM.

Los objetivos que se plantean para el desarrollo del trabajo son:

1. Segmentación semántica de los objetos relevantes de la escena mediante redes neuronales profundas (*deep learning*).

2. Seguimiento de los objetos potencialmente móviles teniendo en cuenta el movimiento estimado de la cámara.

3. Evaluación experimental con diferentes configuraciones de nuestro sistema de SLAM visual en varios datasets públicos.

Los métodos desarrollados en este trabajo han sido optimizados para el dominio específico de la navegación de automóviles. Fuera de este escenario, la estrategia propuesta seguiría siendo válida a alto nivel, pero partes de ella deberían ser adaptadas (por ejemplo, el entrenamiento de la segmentación semántica). En la figura 1.3 se muestra un esquema general del funcionamiento de DOT, con sus entradas y salidas, y posibles campos de aplicación.

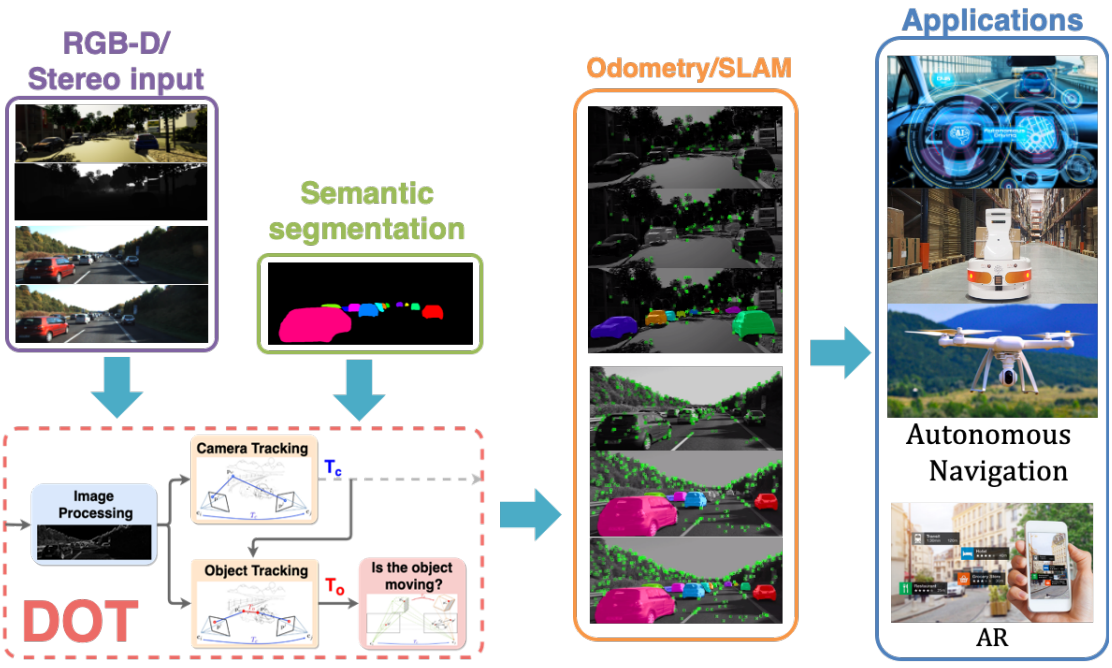

Figura 1.3: Esquema general del sistema DOT y posibles aplicaciones.

1.3. Estructura del documento

El documento se estructura de la siguiente manera:

- En el Capítulo 1 se han presentado la introducción y los objetivos del trabajo.
- En el Capítulo 2 se resume el estado del arte de los métodos de SLAM dinámico y de las técnicas de segmentación semántica.
- En el Capítulo 3 se presenta una visión general del algoritmo desarrollado y se explica en detalle cada una de sus partes.
- El Capítulo 4 recoge la evaluación experimental así como detalles de implementación.
- El Capítulo 5 reúne las principales contribuciones al estado del arte y plantea posibles líneas de continuación del trabajo.
- Finalmente, el Capítulo 6 resume las principales conclusiones.

Capítulo 2

Estado del arte

2.1. Sistemas SLAM

Un sistema de SLAM visual se puede clasificar dentro de los métodos directos o indirectos, en función de la forma en la que utiliza la información proveniente de la imagen y del error que minimiza para obtener la *pose* de la cámara.

En los métodos indirectos, primero se extraen "características de la imagen" (*features*), que después se utilizan para localizar la cámara y crear el mapa. Estas *features* pueden ser elementos geométricos que son fácilmente identificables, como esquinas, o se pueden emplear "descriptores de características" (*feature descriptors*) como ORB, SIFT, FAST... Para determinar la *pose* de la cámara, estos métodos minimizan un error geométrico, calculado como la diferencia en la reproyección en el fotograma de los puntos 3D del mapa. Por el contrario, los métodos directos no realizan esta extracción intermedia de *features*, sino que utilizan directamente la intensidad del píxel para estimar la *pose* de la cámara mediante la minimización del error fotométrico.

Dentro del grupo de los métodos indirectos, se encuentra ORB-SLAM2 [1], un sistema SLAM para cámaras monoculares, estéreo y RGB-D, considerado actualmente uno de los sistemas de referencia del estado del arte. Para asegurar su robustez frente a datos espurios, y en concreto para objetos dinámicos, ORB-SLAM2 [1] utiliza el algoritmo RANSAC. Sin embargo, cuando la escena contiene un gran número de objetos dinámicos este método no es suficiente. En la mayor parte de los casos se produce una pérdida notable de precisión en el cálculo de la trayectoria o incluso el fallo completo del sistema.

2.2. SLAM en entornos dinámicos

Aunque en la actualidad no existe una solución definitiva capaz de lidiar de forma satisfactoria con un entorno dinámico, sí se han ido desarrollando diversos trabajos en

SLAM visual dinámico con diferentes enfoques, que se pueden resumir en tres grandes categorías.

La primera de las categorías, la más general, aspira a plantear un modelo de escena no rígido, incluyendo objetos deformables y dinámicos. Aunque se trata de un planteamiento muy interesante por su generalidad y aplicaciones, plantea también retos significativos. En este trabajo nos limitamos a considerar que el mundo está compuesto por un número variable de sólidos rígidos, que es donde se encuadran las otras dos categorías de SLAM visual dinámico.

La segunda de las categorías de SLAM visual dinámico busca mejorar la precisión y robustez de un sistema SLAM visual permitiendo reconstruir un modelo estático de la escena. Para ello, se segmentan (e ignoran) los objetos dinámicos de la escena con el objetivo de evitar que el algoritmo de tracking use correspondencias que pertenezcan a estos objetos. En esta línea se desarrollan trabajos como DynaSLAM [2], un sistema construido sobre ORB-SLAM2 [1] cuyo objetivo es la construcción de mapas estáticos que puedan reutilizarse en aplicaciones a largo plazo.

La detección de objetos dinámicos se lleva a cabo mediante la combinación de la segmentación semántica para objetos potencialmente móviles y la geometría multi-vista para detección de inconsistencias con el modelo rígido. La segmentación semántica se realiza mediante una red neuronal profunda, Mask R-CNN [6], que detecta y clasifica los objetos de la escena en diferentes categorías, algunas de las cuales han sido preestablecidas como potencialmente móviles (e.g., coche o persona). El método de geometría multi-vista comprueba si un objeto se está moviendo mediante la reproyección de sus puntos 3D en el fotograma actual y la comprobación de la profundidad actual en la imagen RGB-D. El objeto se etiqueta como dinámico si la profundidad reproyectada no coincide con la profundidad medida. Además, DynaSLAM[2] es capaz de completar las partes ocluidas por los objetos en movimiento a partir de la información estática de fotogramas previos.

DynaSLAM fue diseñado para la creación de mapas estáticos por lo que en sus máscaras se incluyen todos los objetos potencialmente móviles de la escena. Esto provoca que, al evaluar las trayectorias estimadas como sistema de SLAM en escenas con objetos potencialmente móviles pero que en realidad no se mueven (p.ej., con muchos coches aparcados), el método proporcione peores resultados respecto a la versión original de ORB-SLAM2 [1]. La razón es que la eliminación de los puntos localizados en los objetos potencialmente móviles repercute negativamente en el proceso de estimación de la trayectoria de la cámara. El objetivo de este trabajo es, precisamente, intentar evitar este problema ya que sólo se etiquetan como dinámicos aquellos objetos que se estén moviendo en ese preciso momento.

Otro trabajo que sigue una línea similar es StaticFusion [10]. En este caso, los autores desarrollan su propio sistema de SLAM visual RGB-D denso y buscan estimar el movimiento de la cámara a la vez que segmentan las partes dinámicas de la escena. En cada fotograma entrante, los componentes de la escena se segmentan en K clusters geométricos y se clasifican como objetos dinámicos aquellos con residuos altos, es decir, si su movimiento no coincide con el de la cámara. El movimiento de la cámara se estima alineando los fotogramas entrantes con el modelo denso y estático del entorno que se ha ido creando a lo largo del procesamiento de la secuencia. Además, StaticFusion utiliza la reconstrucción 3D del fondo de la escena como forma de propagación de la información temporal sobre las partes estáticas de la escena.

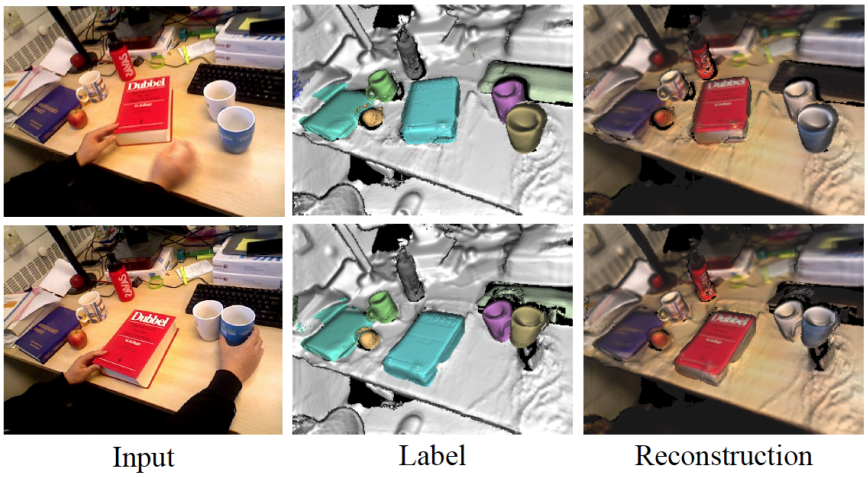

Figura 2.1: Vista general del sistema MID-Fusion [5]. Izq.: Fotogramas entrantes. Centro: Objetos segmentados. Dcha.: Reconstrucción del fondo de la escena y los objetos dinámicos (excepto las manos de la persona).

Por último, la tercera línea de trabajo en SLAM visual dinámico, que va más allá de la segmentación y supresión de objetos dinámicos, es la que siguen trabajos como MID-Fusion [5] (ver figura 2.1) y MaskFusion [3]. Su objetivo es reconstruir el fondo de la escena y estimar también el movimiento de los diferentes objetos dinámicos. Para ello se crean submapas de cada uno de los posibles objetos móviles. MaskFusion [3] emplea *surfels* para la representación de los modelos 3D mientras que MID-Fusion [5] realiza una representación volumétrica basada en *octrees*. Ambos utilizan Mask R-CNN [6] como punto de partida para la segmentación de la escena, aunque luego la refinen mediante técnicas diferentes. La forma de estimar el movimiento de los objetos dinámicos y de la cámara también es diferente: mientras MaskFusion [3] utiliza un método similar al de ElasticFusion [11], MID-Fusion [5] lo hace mediante la minimización de una suma ponderada de los errores geométrico y fotométrico. El error geométrico corresponde al

residuo del algoritmo ICP (Iterative Closest Point) punto-plano. En este algoritmo, una nube de puntos se queda fija mientras se transforma la segunda con el fin de encajarla en la primera y minimizar el error de profundidad entre el mapa de profundidad actual y la proyección del mapa del modelo de referencia. El error fotométrico se calcula a partir de las intensidades de la proyección del mapa de profundidad del modelo de referencia y la del mapa de profundidad actual. Los coeficientes para la combinación de los errores se calculan a partir de la incertidumbre de medida para cada uno de ellos.

Un inconveniente que presentan los sistemas citados que hacen uso del *deep learning* como herramienta de segmentación es que en la actualidad no están implementados en tiempo real debido a las limitadas frecuencias de la redes de segmentación. La contribución desarrollada en este trabajo hace innecesario segmentar todos los fotogramas, lo que permite que el sistema sea independiente de la frecuencia de segmentación de la red y posibilite su implementación en tiempo real.

2.3. Segmentación semántica

En visión por computador, segmentar una imagen significa dividirla en áreas de píxeles que comparten características asignando una etiqueta a cada píxel. El objetivo es obtener una representación más significativa de la imagen para facilitar su análisis o interpretación. En este trabajo, el bloque de segmentación semántica resulta necesario para reconocer los conjuntos de píxeles de los objetos potencialmente móviles, como por ejemplo vehículos en la figura 2.2.

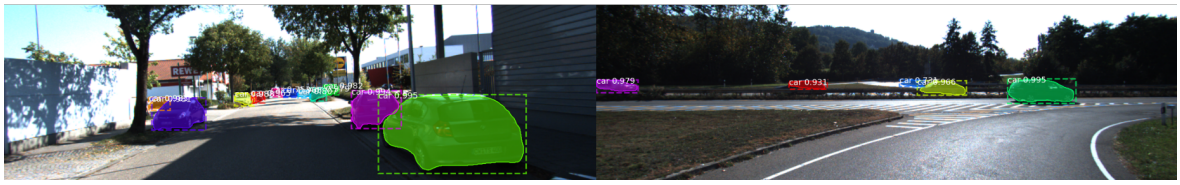

Figura 2.2: Ejemplos de segmentación realizada por Mask R-CNN [6] en el secuencias del *dataset* KITTI [7].

Existen numerosos algoritmos tradicionales de segmentación de imágenes: métodos de agrupamiento (*clustering*), métodos basados en histogramas, de umbralización, etc. A estos métodos más clásicos, y gracias al continuo incremento de potencia computacional, se han sumado recientemente otros muy potentes basados en aprendizaje profundo o *deep learning*. Estos métodos utilizan redes convolucionales (CNN) para llevar a cabo la tarea de segmentación. Las CNN son un tipo de red neuronal especializada en datos

con correlación local, y por eso resultan particularmente útiles para tareas como la clasificación de imágenes.

Una red convolucional se compone de un conjunto de capas de convolución, capas de reducción de muestreo *(pooling layers)* y capas de clasificación (*fully conected layers)*, y está diseñada para aprender automáticamente jerarquías espaciales de características visuales mediante algoritmos de retro-propagación. Las primeras capas de la red detectan características de bajo nivel, como bordes o colores, mientras que las últimas son capaces de identificar características más complejas, como objetos completos o partes de ellos. El trabajo de desarrollo requerido en una CNN es muy inferior al necesario para otros algoritmos de clasificación y, mientras que en los métodos más clásicos los filtros se diseñan a mano con objetivos concretos, si el entrenamiento de la red es suficiente, las CNNs son capaces de identificar y aprender qué filtros y características son más efectivos para sus objetivos de clasificación o segmentación.

Desde el desarrollo de las primeras redes convolucionales (*Convolutional Neural Networks* o *CNN*), se han diseñado nuevas arquitecturas de mayor complejidad para dar solución a diferentes retos de la visión por computador más allá de la clasificación de imágenes. El problema de la detección de objetos (ver figura 2.3), donde no sólo se obtiene la clase de la imagen, sino también la localización espacial de múltiples objetos pertenecientes a diferentes clases, se puede resolver mediante la arquitectura *Region Based Convolution Neural Network (R-CNN)*[12]. Esta red genera recuadros delimitadores (*bounding boxes*) alrededor de cada objeto presente en la imagen como resultado de un proceso de dos etapas: una primera, en la cual se selecciona un cierto número de regiones, y una segunda en la cual se lleva a cabo la clasificación de dichas regiones. La selección de las regiones se realiza mediante un algoritmo de búsqueda selectiva de regiones de interés, limitado a 2000 de ellas. La clasificación se lleva a cabo mediante una CNN que extrae vectores de características para cada región y que, posteriormente, sirven como entrada a un conjunto de máquinas de vectores de soporte *(SVMs)* que devuelven como resultado la clase de la región. El principal problema de R-CNN es que su entrenamiento requiere una gran cantidad de tiempo y no puede ser utilizada para problemas en tiempo real. Con el objetivo de reducir el tiempo de procesamiento, la arquitectura Fast R-CNN [13] propone utilizar como base R-CNN, pero compartir la computación de las capas convolucionales entre las diferentes regiones seleccionadas. Para ello, añade una CNN en la entrada de la arquitectura que genera un mapa convolucional de características, a partir del cual se seleccionan las regiones, a diferencia de R-CNN donde se seleccionan a partir de la imagen original. Para reducir todavía más el tiempo de computación, la arquitectura Faster R-CNN [14] propone utilizar una *region proposal network (RPN)* en lugar de un algoritmo de búsqueda

selectiva para optimizar la selección de las regiones, eligiendo aquellas en las que la red predice la presencia de objetos.

En este trabajo se ha decidido utilizar Mask R-CNN [15], una extensión de la arquitectura Faster R-CNN [14], que además de proporcionar como resultado la categoría de cada objeto detectado en la imagen y las coordenadas de su recuadro delimitador (*bounding box*), segmenta cada píxel perteneciente al objeto. Es lo que se denomina en la literatura como *instance segmentation* y constituye una gran ventaja respecto al uso de otros métodos de segmentación, porque permite simplificar significativamente el algoritmo de seguimiento de los objetos, en especial cuando ciertos objetos de una misma categoría se superponen entre ellos.

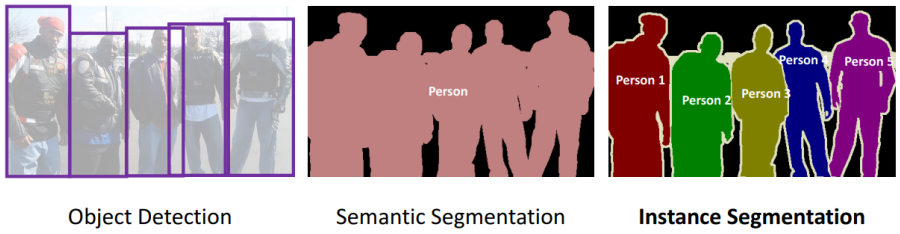

Figura 2.3: Izq.: Detección de objetos. Centro: Segmentación semántica. Dcha: Segmentación de objetos individuales (*instance segmentation*). Fuente: "Semantic Segmentation Techniques For Computer Vision Tasks" [8]

La técnica de *instance segmentation* permite segmentar la imagen en objetos individuales: además de identificar los píxeles que pertenecen a una categoría, marca por separado los distintos elementos de dicha categoría presentes en una imagen. Para ello, primero realiza la detección de objetos con el fin de extraer los *bounding boxes* alrededor de cada objeto, como en detección de objetos, y después lleva a cabo la segmentación binaria dentro de cada recuadro para separar el objeto del fondo de la imagen. La figura 2.4 muestra la arquitectura de Mask R-CNN [6], que se describe con más detalle a continuación.

Al igual que Faster R-CNN [14], se compone de una primera capa formada por una red neuronal convolucional que permite extraer los mapas de características de la imagen. Después, tomando como entrada estos mapas de características, se aplica una *region proposal network (RPN)* con el objetivo de seleccionar un conjunto de regiones en las que la red predice la posible existencia de objetos. La siguiente componente es una capa de reducción de muestreo (*pooling layer*), que asegura que todas las regiones tengan el mismo tamaño y, posteriormente, estas regiones pasan a través de una *fully connected network* que clasifica los objetos y genera su recuadro delimitador en la imagen.

Hasta aquí las etapas descritas se corresponden con la arquitectura de Faster R-

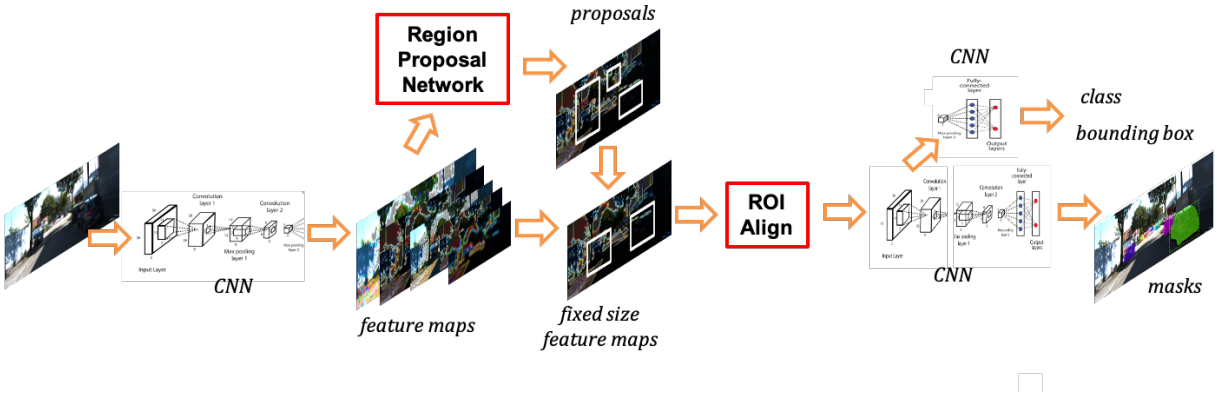

Figura 2.4: Diagrama de la arquitectura de Mask R-CNN [6].

CNN [14], pero Mask R-CNN [6] incluye una rama en paralelo para generar máscaras que codifican la localización de los píxeles de cada objeto. Para ello, y con el objetivo de reducir el tiempo de computación, primero se seleccionan las regiones de interés *(ROI)* como aquellas cuya área de intersección con el resto de las regiones supone al menos la mitad de su área. Por último, se emplea una red convolucional que toma como entrada las regiones seleccionadas por el clasificador ROI y genera las máscaras para cada uno de los objetos, obteniendo resultados como los que se han presentado en la figura 2.2.

Capítulo 3

Fundamentos de "Dynamic Object Tracking" (DOT)

"Dynamic Object Tracking" (DOT) es el sistema desarrollado en este trabajo para la identificación y seguimiento de objetos dinámicos a partir de una secuencia de imágenes. El algoritmo recibe como entrada una serie de fotogramas RGB, sus correspondientes imágenes de profundidad y la instanciación semántica de los objetos dinámicos obtenida por la red neuronal. A partir de esta información DOT estima el movimiento de la cámara y de los objetos, y mediante cálculos geométricos determina qué objetos se encuentran efectivamente en movimiento. Finalmente, se actualizan las máscaras que distinguen las partes estáticas de las dinámicas y que pueden alimentar a su vez a un sistema de odometría visual o SLAM.

3.1. Diagrama y estructura

La figura 3.1 muestra una visión global mediante un diagrama de bloques del funcionamiento del algoritmo.

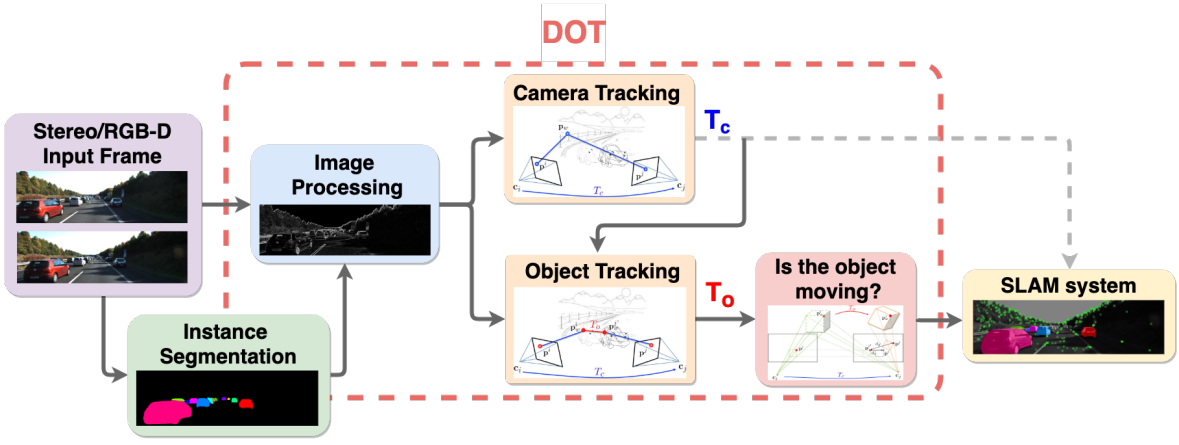

Figura 3.1: Diagrama de bloques del funcionamiento de DOT.

DOT utiliza imágenes RGB-D (una imagen en color junto a una matriz con las profundidades de los píxeles) o estéreo (un par de imágenes tomadas instantáneamente por cámaras separadas una distancia conocida) a una cierta frecuencia de vídeo.

El primer bloque (*Instance Segmentation*) corresponde a la red neuronal que lleva a cabo la segmentación semántica de la escena identificando mediante máscaras las partes de la imagen que corresponden a objetos dinámicos (en nuestra parte experimental corresponderán sólo a vehículos). Como se explica más adelante, la frecuencia a la que opera la red no tiene por qué ser la del vídeo, sino que puede ser menor.

El bloque de procesamiento de imágenes (*Image processing*) extrae y separa los puntos que pertenecen a regiones estáticas de la imagen y los puntos que se encuentran en objetos dinámicos. El movimento de la cámara entre fotogramas (*Camera tracking*) se estima utilizando únicamente la parte estática de la escena. A partir de este bloque, y teniendo en cuenta el movimiento de la cámara, se calcula de forma independiente el movimiento de cada uno de los objetos segmentados por la red (*Object tracking*).

El último bloque (*Is the object moving?*) determina, utilizando la geometría, si los objetos que habían sido etiquetados como dinámicos por la red están efectivamente en movimiento o por el contrario se encuentran estáticos. Con esta información se actualizan las máscaras que codifican las regiones estáticas y dinámicas de cada fotograma y que pueden alimentar la entrada de un sistema de odometría/SLAM visual.

3.2. Segmentación semántica

La alimentación del bloque de segmentación es la imagen RGB original. Ajustamos la red para obtener como resultado una única máscara con todos los objetos potencialmente móviles segmentados. Todo el resto del área, que la red no ha clasificado en una de sus categorías, se etiqueta como "fondo de la imagen" y se considera permanentemente estática en los bloques posteriores.

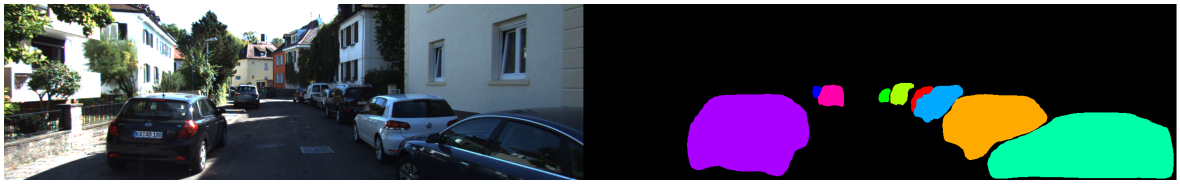

Figura 3.2: Izda.: Imagen RGB utilizada como entrada. Dcha.: Máscara de segmentación obtenida como resultado de Mask R-CNN [6], donde aparecen en color los objetos potencialmente móviles que han sido segmentados.

La implementación de Mask R-CNN [6] permite utilizar los coeficientes obtenidos al entrenar la red con el *dataset* MS COCO [16], que incluye 91 categorías de objetos.

Como el objetivo de este trabajo es seguir el movimiento de una serie de objetos, se han restringido las categorías a aquellas consideradas potencialmente móviles. Si fuera necesario incluir otras categorías, se podría ajustar la red utilizando estos coeficientes como punto de partida o entrenarla desde cero con un *dataset* propio.

3.3. Estimación del movimiento de la cámara y de objetos dinámicos

El objetivo del algoritmo es estimar la posición y orientación de una serie de objetos a lo largo del tiempo y determinar cuándo se están moviendo. Para ello, primero se calcula la posición y orientación de la cámara y después el movimiento de los objetos.

3.3.1. Procesamiento de la imagen

A partir de la segmentación semántica podemos separar los puntos que pertenecen al "fondo de la imagen" (**puntos estáticos**) y los que se encuentran en objetos potencialmente móviles (**puntos dinámicos**). Sin embargo, no todos los píxeles de la imagen son útiles para la estimación del movimiento, sino solamente los que presentan un elevado gradiente fotométrico. La figura 3.3 muestra a la izquierda la imagen de gradientes fotométricos y a la derecha un ejemplo de los puntos de alto gradiente seleccionados.

Figura 3.3: Izq.: Imagen del gradiente fotométrico. Dcha.: Ejemplo de la selección de puntos de alto gradiente de la región estática de la imagen.

3.3.2. Proyección multivista de puntos estáticos y dinámicos

Modelo de proyección multivista de puntos estáticos: Para un sistema de SLAM visual que opera en un ambiente estático, donde todos los elementos permanecen inmóviles, el cambio de posición de un punto en diferentes fotogramas se debe únicamente al movimiento de la cámara. Si la escena es estática y se conocen la calibración de la cámara y las correspondencias entre puntos, es posible calcular el movimiento de la cámara utilizando ecuaciones geométricas multivista [17].

La figura 3.4 representa la cámara en dos instantes de tiempo j e i, identificados en la figura como c_j y c_i y la proyección de un punto fijo $\mathbf{p}$ en ambos fotogramas. El

modelo que se explica a continuación relaciona las coordenadas de los píxeles del punto $\mathbf{p}$ en cada fotograma (u, v) con el movimiento relativo de la cámara $\mathbf{T}_c$.

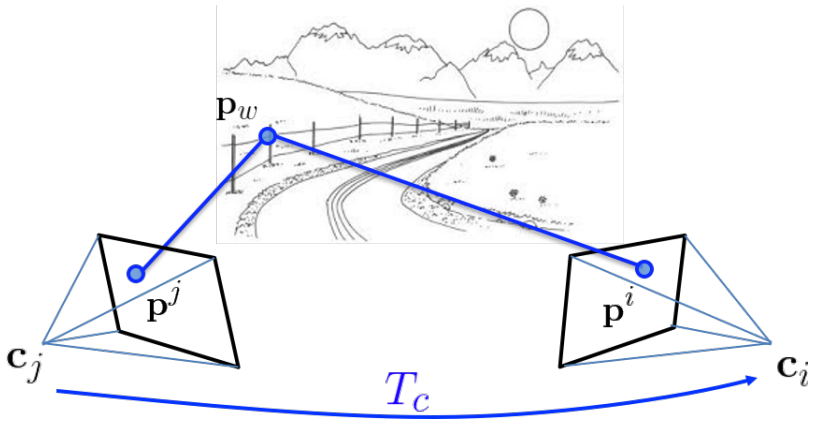

Figura 3.4: Modelo de proyección de un punto estático.

El diagrama de la figura 3.5 muestra el proceso de proyección de un punto desde sus coordenadas en la imagen I_j a las coordenadas correspondientes en la imagen I_i, conocidas la calibración de la cámara $\mathbf{K}$ (ver anexo A.2), el modelo de proyección Π (cámara estenopeica,"*pinhole*" en nuestro caso) y la profundidad del punto z_c.

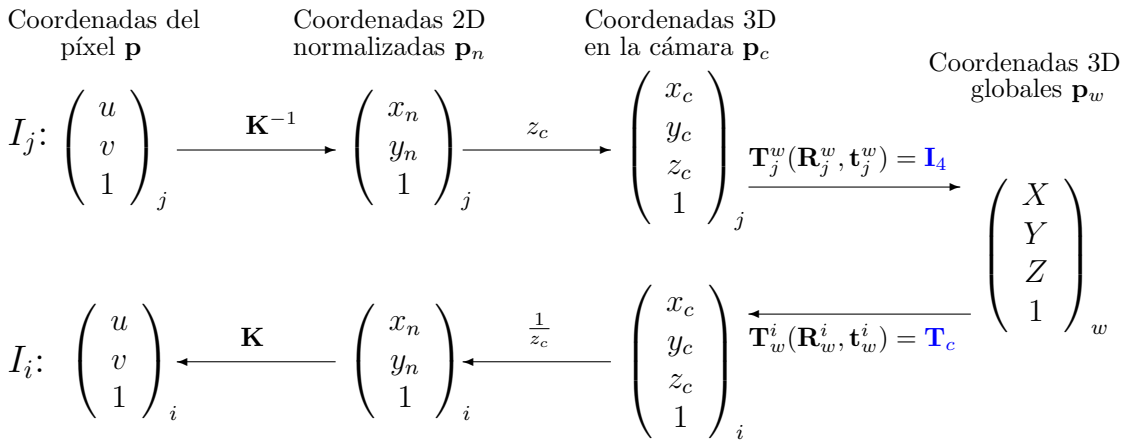

Figura 3.5: Diagrama del proceso de proyección de un punto estático en la imagen entre fotogramas tomados en los instantes j e i.

A partir de las coordenadas expresadas en píxeles de un punto $\mathbf{p}$ en la imagen la transformación a coordenadas 2D normalizadas $\mathbf{p}_n$ se obtiene mediante la matriz de calibración $\mathbf{K}$:

$$\mathbf{p} = \mathbf{K}\mathbf{p}_n. \tag{3.1}$$

Las coordenadas de un punto en el sistema de referencia de la cámara $\mathbf{p}_n$ normalizadas por su profundidad z_c son una representación de todos los puntos 3D que se

encuentran en el mismo rayo de proyección:

$$\mathbf{p}_c = z_c \mathbf{p}_n. \tag{3.2}$$

La relación entre las coordenadas 3D de un punto expresadas en el sistema de referencia de la cámara y el sistema de referencia absoluta viene dada por las ecuaciones del sólido rígido:

$$\mathbf{p}_w = \mathbf{R}_c^w \mathbf{p}_c + \mathbf{t}_c^w, \tag{3.3}$$

donde $\mathbf{R} \in SO(3)$ y $\mathbf{t} \in \mathbb{R}^3$ son las matrices de rotación y traslación (ver anexo A.1). Los subíndices w y c indican que la transformación se realiza desde el sistema de coordenadas centrado en la cámara al sistema global (*world*). Por simplicidad se puede formular la transformación del sólido rígido en coordenadas homogéneas:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{3.4}$$

Mediante la composición de matrices expresamos el movimiento de una cámara respecto de la otra $\mathbf{T}_j^i$. Dado que el objetivo es calcular el movimiento relativo de la camára entre imágenes para poder identificar objetos dinámicos, hacemos coincidir convenientemente el sistema de referencia absoluta con el de la imagen de referencia j ($\mathbf{T}_j^w = \mathbf{I}_4$). De este modo y a partir de aquí nos referiremos al movimiento de la cámara $\mathbf{T}_c$ como:

$$\mathbf{T}_c = \mathbf{T}_j^i = \mathbf{T}_w^i \mathbf{T}_j^w = \mathbf{T}_w^i \mathbf{I}_4 = \mathbf{T}_w^i. \tag{3.5}$$

Llamamos Π a las transformaciones necesarias para proyectar las coordenadas de un punto 3D en el sistema de referencia de la cámara a la imagen:

$$\mathbf{p} = \Pi(\mathbf{p}_c), \qquad \mathbf{p}_c = \Pi^{-1}(\mathbf{p}, z_c). \tag{3.6}$$

Con lo que, finalmente, podemos expresar la proyección de un punto de una imagen a otra como:

$$\mathbf{p}^i = \Pi(\mathbf{T}_c \Pi^{-1}(\mathbf{p}^j, z_j)). \tag{3.7}$$

Modelo de proyección multivista de puntos dinámicos: Hasta este punto, la formulación es válida siempre y cuando se cumpla la condición de estaticidad de la escena y el único movimiento sea el de la cámara. Como se muestra en la figura 3.6 (compárese con la figura 3.4), si existen objetos que se mueven de forma independiente, no es posible determinar a priori qué cambios en la imagen son debidos al movimiento

de la cámara y cuáles al movimiento de los objetos, ya que ambos aparecen acoplados en las imágenes.

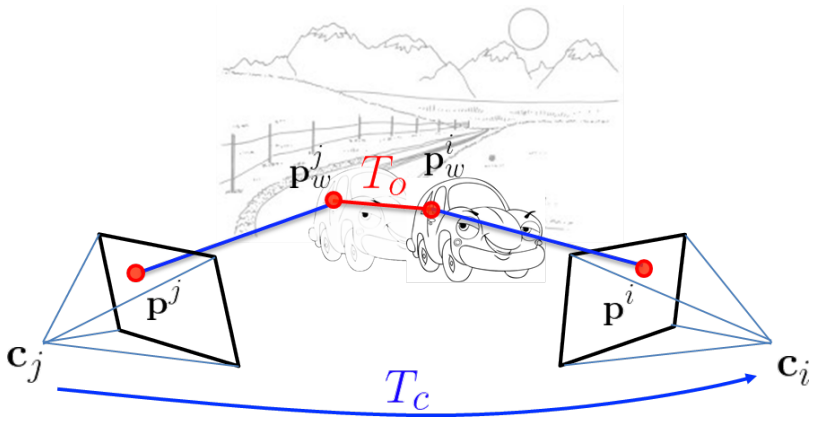

Figura 3.6: Modelo de proyección de un punto dinámico.

Sin embargo, ya que gracias a la segmentación semántica podemos distinguir las partes estáticas de la escena, es posible calcular el movimiento de la cámara $\mathbf{T_c}$ utilizando los puntos estáticos y a continuación sustraerlo, para estimar el movimiento de cada objeto $\mathbf{T}_o$ utilizando los puntos que los conforman.

El modelo que permite relacionar las coordenadas de un punto perteneciente a un objeto en movimiento entre dos fotogramas es una extensión del explicado previamente, con la diferencia de que ahora es necesario tener en cuenta que el punto $\widetilde{\mathbf{p}}_w$ ya no permanece inmóvil ($\widetilde{\mathbf{p}}_w^j \neq \widetilde{\mathbf{p}}_w^i$). El diagrama de la figura 3.7 muestra el proceso de proyección de un punto perteneciente a un objeto en movimiento.

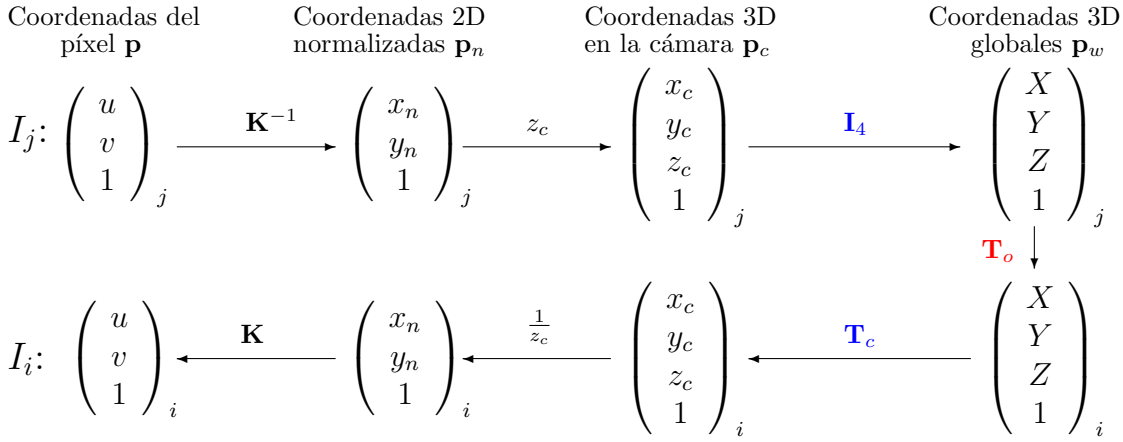

Figura 3.7: Diagrama del proceso de proyección de un punto dinámico en la imagen entre fotogramas tomados en los instantes j e i.

Las relaciones entre los diferentes sistemas de coordenadas son idénticas a las descritas para el caso en el que el punto fuese estático, salvo que ahora consideramos el

movimiento como sólido rígido del objeto $\mathbf{T}_o$ expresado como

$$\widetilde{\mathbf{p}}_w^i = \mathbf{R}_o \widetilde{\mathbf{p}}_w^j + \mathbf{t}_o. \tag{3.8}$$

Utilizando coordenadas homogéneas y, al igual que en el apartado anterior, y haciendo coincidir el origen de coordenadas global con el centro óptico de la cámara en el instante j, la ecuación que relaciona un punto entre instantes expresado en coordenadas de la cámara se puede reescribir como:

$$\widetilde{\mathbf{p}}_c^i = \mathbf{T}_c \mathbf{T}_o \widetilde{\mathbf{p}}_c^j, \tag{3.9}$$

donde

$$\mathbf{T}_o = \begin{bmatrix} \mathbf{R}_o & \mathbf{t}_o \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{3.10}$$

De la misma forma que con los puntos estáticos, puede formularse la proyección de un punto dinámico entre dos imágenes i y j, que en este caso queda como:

$$\widetilde{\mathbf{p}}^i = \Pi(\mathbf{T}_c \mathbf{T}_o \Pi^{-1}(\widetilde{\mathbf{p}}^j, z_j)). \tag{3.11}$$

3.3.3. Errores de reproyección

El cálculo del movimiento de la cámara es un problema no lineal que se plantea en términos de minimización del error de reproyección E:

$$\mathbf{T}_c(\mathbf{R}_c, \mathbf{t}_c) = \mathbf{argmin}(E), \tag{3.12}$$

o, en otras palabras, la rotación y traslación que ha experimentado la cámara son aquéllas que sitúan las proyecciones de los puntos en sus correspondencias adecuadas entre imágenes.

Los enfoques basados en *features* consideran un error geométrico en la imagen, calculado como la diferencia de las coordenadas en píxeles entre el punto reproyectado y su correspondencia en la imagen, ponderados mediante una norma robusta γ (veáse sección 3.3.5):

$$E_g = \sum_{\mathbf{p} \in P} |\mathbf{p}^j - \mathbf{p}^i|_\gamma. \tag{3.13}$$

En algunos trabajos más recientes se utilizan "métodos directos", que calculan un error fotométrico a partir de la diferencia entre las intensidades de los píxeles con un alto gradiente lumínico:

$$E_f = \sum_{\mathbf{p} \in P} |I_j(\mathbf{p}^j) - I_i(\mathbf{p}^i)|_\gamma. \tag{3.14}$$

Ambas soluciones plantean ventajas e inconvenientes, e incluso podría darse la posibilidad de que las odometrías visuales y el SLAM se beneficiasen de la combinación de ambas. En este trabajo hemos utilizado el error fotométrico tanto para el cálculo de la posición de la cámara como para la posición de los objetos. A continuación se discuten brevemente las principales diferencias entre ambos métodos.

Métodos geométricos/indirectos: Los métodos indirectos necesitan un proceso previo a la optimización, para la extracción de características (*features*) y búsqueda de correspondencias de los puntos entre las imágenes. Este proceso no sólo añade tiempo de computación sino que, si no se hace correctamente, puede ser una fuente de emparejamientos espurios que empeoran el funcionamiento del algoritmo. Por otra lado, esta etapa de preprocesamiento permite optimizaciones para movimientos más acusados de las cámaras y reduce los errores geométricos de modelado. Además, en los sistemas de SLAM facilita los procesos de relocalización y cierre de bucle, fundamentales para una navegación robusta y sin deriva.

Métodos fotométricos/directos: Los métodos directos no requieren ningún procesamiento previo, puesto que alinean "directamente" las intensidades lumínicas de los píxeles [18]. Los errores fotométricos han demostrado ser en general más robustos en los procesos de cálculo del movimiento de la cámara, ya que acceden a más información al muestrear píxeles de todas las regiones de la imagen, incluyendo áreas con variaciones sutiles de intensidad donde resulta difícil extraer *features*. Sin embargo, los procesos de optimización y relocalización resultan más costosos y complejos y son menos robustos frente a errores geométricos de modelado.

Puesto que el objetivo de este trabajo es el cálculo del movimiento de la cámara en breves períodos de tiempo, y teniendo en cuenta que es necesario utilizar píxeles en toda la imagen para poder determinar el movimiento de los objetos y que DOT no se beneficia por el momento de módulos como la relocalización o el cierre de bucle, el método de optimización elegido ha sido el fotométrico. Nótese que aunque escapa al contenido de este trabajo, el sistema se beneficiaría de la combinación con *features* especialmente en presencia de movimientos acusados de los objetos.

Algunos métodos directos utilizan todos los píxeles de la imagen, creando así una reconstrucción densa de la escena. Pero, dado que en esta aplicación el objetivo no es la generación de un mapa sino el cálculo del movimiento, y que esto resultaría muy costoso y poco eficaz, trabajaremos siempre con una selección de los puntos de alto gradiente lumínico tratando de tener una distribución dispersa y uniforme mediante la división de la imagen en regiones.

3.3.4. Movimiento de la cámara y de los objetos dinámicos

Como se ha discutido en secciones anteriores, el cálculo del movimiento de los objetos dinámicos se realiza en una segunda etapa tras la estimación del movimiento de la cámara a partir de los puntos estáticos. En ambos casos se trata de resolver un problema de optimización no lineal equivalente, en el que las incógnitas son la traslación y la rotación de la cámara o del objeto y la función de coste es la suma de los errores fotométricos de reproyección.

De esta forma, el problema de cálculo del movimiento de la cámara queda formulado en términos de la función de proyección de los puntos estáticos (ver ecuación 3.7):

$$\mathbf{T}_c(\mathbf{R}_c, \mathbf{t}_c) = \mathbf{argmin} \sum_{\mathbf{p} \in P} |I_j(\mathbf{p}^j) - I_i(\Pi((\mathbf{T}_c \Pi^{-1}(\mathbf{p}^j, z_j)))|_\gamma. \tag{3.15}$$

Una vez hallada la matriz de transformación de la cámara $\mathbf{T_c}$, se estima por separado la posición y orientación de cada uno de los objetos potencialmente móviles utilizando los puntos que los componen. El error fotométrico en este caso se basa en las relaciones de proyección de puntos dinámicos definidas en la ecuación 3.11:

$$\mathbf{T}_o(\mathbf{R}_o, \mathbf{t}_o) = \mathbf{argmin} \sum_{\mathbf{p} \in P} |I_j(\mathbf{p}^j) - I_i(\Pi(\mathbf{T}_c \mathbf{T}_o \Pi^{-1}(\widetilde{\mathbf{p}}^j, z_j)))|_\gamma. \tag{3.16}$$

3.3.5. Método de optimización

La resolución del problema no lineal del movimiento de la cámara suele resolverse con pequeñas variaciones del método iterativo de optimización de Gauss-Newton. En este trabajo hemos utilizado una implementación preexistente del problema con la librería *Ceres*. *Ceres* ofrece una solución en C++ versátil y eficiente para la resolución de problemas no lineales que incluye, entre otras herramientas, el cálculo de los jacobianos por autodiferenciación del error. Sin embargo, para hacer el algoritmo más rápido y robusto utilizamos una implementación que incorpora los jacobianos calculados analíticamente.

Las opciones adoptadas en este trabajo para modelar el problema con *Ceres* son las siguientes:

- El método iterativo seleccionado para la resolución se encuadra dentro de los denominados "métodos de región de confianza". Estos métodos siguen una estrategia de globalización que busca limitar la influencia de la inicialización del problema en la solución final favoreciendo la convergencia global.

– Elegimos una función de coste robusto de Cauchy [19] que minimiza el efecto de los datos espurios en la solución.

– Para aumentar la robustez empleamos una representación de puntos basada en parches de píxeles. De esta forma, los puntos en las imágenes están compuestos por un conjunto de píxeles adyacentes.

– Con el objetivo de mejorar la convergencia, se ha empleado una estrategia basada en la representación piramidal de resoluciones de la imagen. La optimización comienza utilizando la resolución más baja de la imagen y, una vez la solución converge, se aumenta progresivamente la resolución utilizando como semilla inicial la solución del paso anterior hasta alcanzar la resolución original de la imagen y, por tanto, la solución del problema.

3.4. Detección de objetos en movimiento

El último bloque recibe como entrada el movimiento de la cámara $\mathbf{T}_c$ y las matrices de transformación de los objetos $\mathbf{T}_o$, estima si se encuentran en movimiento o estáticos y crea las máscaras que servirán como alimentación del sistema de SLAM visual acoplado.

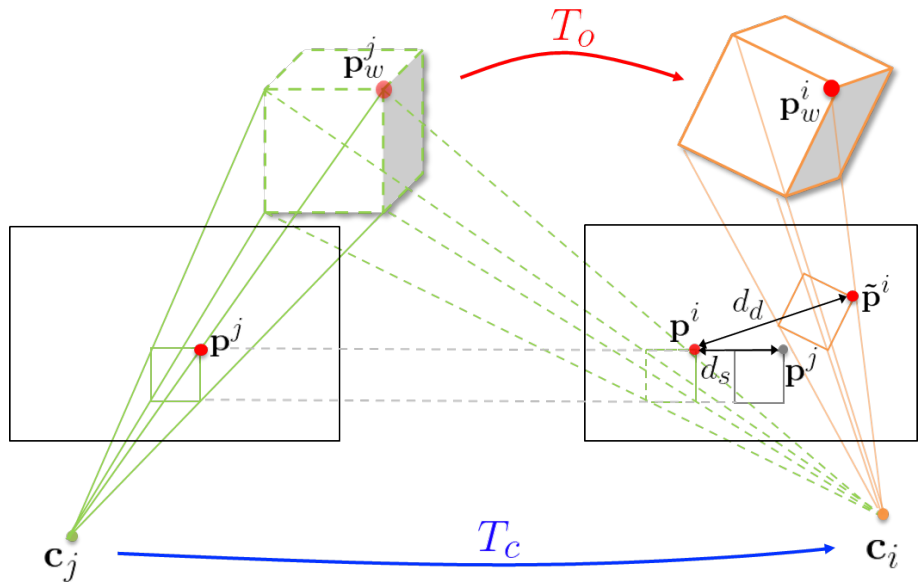

Figura 3.8: Diagrama de proyección de un punto perteneciente a un objeto dinámico en dos fotogramas diferentes j e i.

Una primera aproximación al problema podría considerar que cualquier desviación respecto de la matriz identidad de la matriz de transformación del objeto T_o indicaría que éste se está moviendo. Sin embargo, en la práctica los resultados obtenidos después del proceso de optimización están afectados por altos niveles de incertidumbre,

lo que hace muy difícil distinguir qué parte de las variaciones es debida al ruido en los resultados y cuál a un cambio en la posición del objeto. En este trabajo optamos por determinar el movimiento de los objetos utilizando mediciones 2D en las imágenes, ya que éstas se caracterizan por ser mucho menos ruidosas que los movimientos equivalentes de los puntos en el espacio 3D. La figura 3.8 muestra las proyecciones de un objeto si estuviera estático (color verde) o si, por el contrario, estuviera en movimiento (naranja).

Para **determinar si un objeto se encuentra en movimiento** utilizamos una métrica que se ha denominado disparidad dinámica

$$d_d = \left|\left|\mathbf{p}^i, \widetilde{\mathbf{p}}^i\right|\right|, \tag{3.17}$$

y que expresado con palabras es la distancia entre la proyección del punto si se encontrara estático y la proyección real que obtenemos. Si la mediana de las disparidades dinámicas de todos los puntos de un objeto es mayor que un cierto umbral θ_d determinamos que el objeto se está moviendo. Conviene precisar que esta definición de disparidad no corresponde exactamente con el término normalmente utilizado en visión por computador para el procesado de imágenes estéreo.

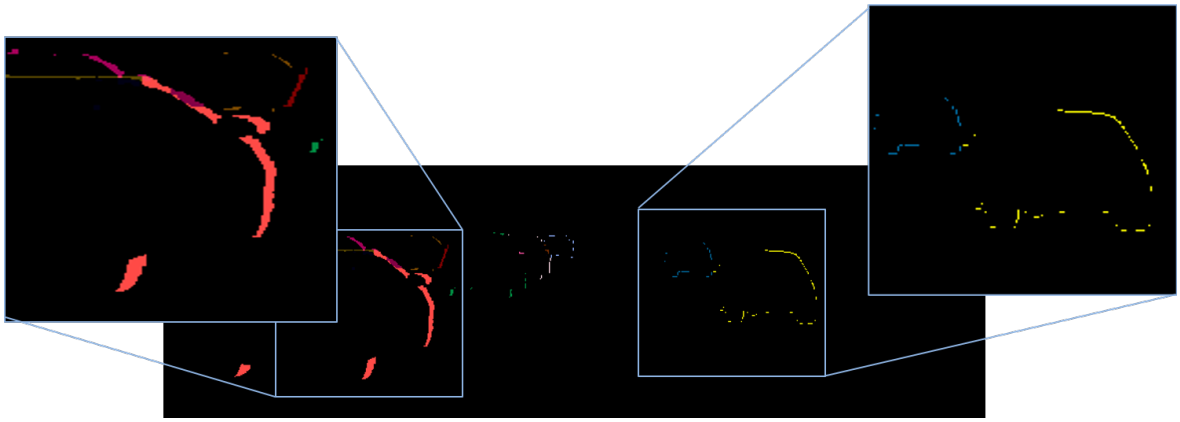

Figura 3.9: Ejemplo de disparidades dinámicas entre dos fotogramas consecutivos.

Si un objeto se encuentra en movimiento, pero éste se traduce en una disparidad dinámica subpíxel, el algoritmo lo etiquetaría erróneamente como estático. Esta restricción afecta a los objetos dependiendo de 1) la posición del punto en la imagen, 2) su profundidad y 3) el ángulo relativo entre las direcciones del movimiento del objeto y de la cámara. Un ejemplo de este efecto puede apreciarse en la figura 3.9: 1) cuanto más centrado se encuentre el objeto en la imagen, 2) más lejos esté de la cámara y 3) menor sea el ángulo entre el movimiento del objeto y la cámara, existen más probabilidades de que, para un mismo cambio de posición, la disparidad esté por debajo del píxel, y por

lo tanto, menos probable sea detectar este movimiento. Para poder **determinar si un objeto se encuentra estático**, su movimiento relativo con la cámara ha tenido que ser el adecuado para que las tres posibles direcciones de traslación sean observables. Para ello calculamos el movimiento en la imagen debido al movimiento relativo con la cámara $(\Delta x_c, \Delta y_c, \Delta z_c)$:

$$d_x = f_x \frac{\Delta x_c}{z_j}, \qquad d_y = f_y \frac{\Delta y_c}{z_j}, \qquad d_z = \frac{\Delta z_c}{z_j(z_j + \Delta z_c)} \sqrt{(f_x x_j)^2 + (f_y y_j)^2}. \tag{3.18}$$

Si el valor que llamamos disparidad estática $d_s = \min(d_x, d_y, d_z)$ supera un cierto umbral θ_s podemos decir que la estaticidad del objeto es observable y que, por lo tanto, si la mediana de disparidades dinámicas de todos los puntos es menor que un cierto valor θ_d el objeto no se está moviendo.

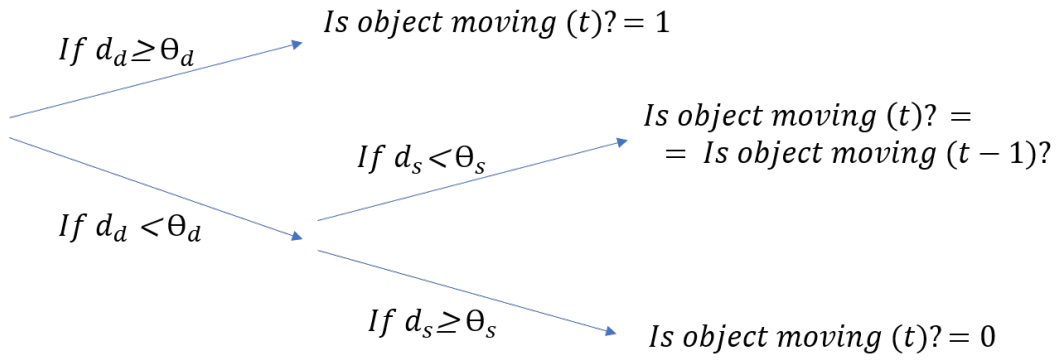

Figura 3.10: Diagrama de relaciones lógicas del procedimiento de detección de los objetos en movimiento.

Un resumen de estas consideraciones se encuentra en la figura 3.10. En el caso particular de que en un cierto instante t no podamos determinar el movimiento del objeto, pero en el que ya hayamos realizado observaciones anteriores, entonces propagamos la observación inmediatamente previa.

En principio, el valor mínimo que debería tener el umbral θ_d sería de 1 píxel, pero en la práctica hay que añadir una última consideración. Dado que la estimación del movimiento de la cámara o de los objetos tiene asociada cierta incertidumbre, ésta debe tenerse también en cuenta a la hora de fijar los umbrales. De esta forma, θ_d debería ser igual a 1 píxel cuando dichas incertidumbres son despreciables pero debería crecer continuamente conforme éstas aumentan. La precisión en las estimaciones del movimiento puede cuantificarse en términos de los coeficientes de correlación de Pearson para la cámara, ϕ_c, y los objetos, ϕ_o. Estos coeficientes, que pueden variar entre 0 y 1, expresan el grado de correlación lineal entre las intensidades de los puntos de referencia y sus correspondientes estimados en un instante posterior (1 y 0 indicarían

relación lineal perfecta y nula, respectivamente). La función adoptada en este trabajo para relacionar θ_d con estos coeficientes ha sido la siguiente:

$$\theta_d = \frac{1}{\phi_c \phi_o} \tag{3.19}$$

Si bien elegir la forma funcional óptima requeriría un estudio más extenso, esta expresión cumple los requisitos establecidos (aumento gradual desde 1 hasta infinito conforme se reducen los coeficientes) y ha demostrado buenos resultados en este trabajo. Cabe destacar que este método ofrece un comportamiento robusto frente a cambios en la iluminación respecto al fotograma de referencia, que podrían invalidar otras alternativas que se basaran en análisis de intensidades, pero que en principio no afectaría a los coeficientes de correlación.

Por tanto, para etiquetar un objeto como estático se exige que la mediana de la disparidad sea menor que un píxel cuando la estimación del movimiento es muy precisa, pero este límite se relaja conforme la incertidumbre asociada aumenta.

La figura 3.11 es un ejemplo de la máscara que propaga DOT. Los objetos etiquetados como "en movimiento" se representan en colores, mientras que los que se ha marcado como "estáticos" desaparecen en color negro. Los coches representados en gris son aquellos en los que se concluye que "no se puede afirmar que permanezcan inmóviles".

Figura 3.11: Ejemplo de la máscara resultante calculada por DOT.

Capítulo 4

Implementación y resultados

4.1. Implementación

La implementación de DOT se ha realizado en C++ utilizando las librerías *Eigen*, *OpenCV* y *Ceres-Solver*. *Eigen* es una librería de C++ para álgebra lineal, operaciones de matrices y vectores, transformaciones geométricas y solucionadores numéricos (pública con licencia de *Mozilla Public License 2.0*). *OpenCV* es la librería multiplataforma más popular para visión por computador (código libre, distribuida bajo licencia BSD). *Ceres-Solver* se ha utilizado para el modelado y la resolución del problema de optimización de la posición de la cámara y los objetos (licencia BSD).

Para la segmentación semántica de las imágenes se ha utilizado una implementación de Mask R-CNN [15] para Python 3, Keras (biblioteca de código abierto de Redes Neuronales distribuida bajo licencia MIT) y TensorFlow (biblioteca para aprendizaje automático con licencia de código abierto Apache 2.0.).

4.2. Evaluación experimental

Descripción del experimento. A pesar de que las aplicaciones potenciales de DOT cubren un amplio espectro que abarca desde la detección de objetos hasta la realidad aumentada o la conducción autónoma entre otros, en este trabajo se ha realizado una evaluación intensiva para demostrar en qué medida "conocer el movimiento de los objetos" mejora la precisión de un sistema SLAM. El experimento consiste en estimar las trayectorias de una cámara con un sistema SLAM de referencia ajustado con tres configuraciones diferentes y evaluar si, efectivamente, aquélla en la que se ha implementado DOT proporciona los mejores resultados tanto en precisión como robustez.

Sistema SLAM. El estudio se ha realizado utilizando en todos los casos ORB-SLAM2 [1], un sistema SLAM representativo del estado del arte y frecuentemente

elegido como referencia. ORB-SLAM2 para cámaras estéreo o RGB-D está construido sobre ORB-SLAM monocular basado en *features* [20], un sistema completo para cámaras monoculares que incluye capacidades de reutilización del mapa, cierre de bucle y relocalización. ORB-SLAM2 proporciona un soporte robusto para llevar a cabo este experimento.

KITTI *Dataset.* Para la evaluación se han seleccionado tres *datasets* públicos diseñados para investigación en conducción autónoma. Dos de ellos forman parte del *dataset* KITTI [7], que contiene secuencias estéreo de escenas urbanas y de carretera grabadas desde un coche. Los datos de las poses reales se han obtenido mediante un sistema de localización GPS. El tercer *dataset* utilizado para la evaluación es V-KITTI [21][9], un *dataset* sintético compuesto por 5 secuencias clonadas virtualmente del *dataset* KITTI [7]. En el Anexo B se adjunta una descripción más detallada de los *datasets* seleccionados.

Configuraciones. Las tres configuraciones elegidas para evaluar las mejoras que introduce DOT al funcionamiento del SLAM son:

- *No masks*: El sistema SLAM funciona con su implementación original sobre las imágenes sin modificar. Se trata en definitiva de suponer que la escena es estática, y se traduce en que todos los puntos de las imágenes (incluidos los que pertenecen a objetos móviles) son susceptibles de ser seleccionados por ORB-SLAM2 para estimar el movimiento de la cámara.

- *DOT masks*: Es el método desarrollado en este trabajo, en el cual el sistema de SLAM recibe como entrada, además de las imágenes, las instancias generadas por la red y modificadas por DOT para detectar los coches en movimiento. De esta forma ORB-SLAM2 no extrae puntos de los objetos que cumplen a la vez haber sido etiquetados como dinámicos por la red y encontrarse en movimiento.

- *All Masks*: Este método provee al SLAM de todas las máscaras que directamente ha calculado la red neuronal sin verificar que los objetos segmentados como potencialmente móviles se encuentran efectivamente en movimiento. De esta forma se eliminan todos los objetos potencialmente móviles.

La figura 4.1 muestra el funcionamiento de ORB-SLAM2 en las tres configuraciones presentadas para dos secuencias de características diferentes. En la fila superior se representa una escena donde la mayoría de los vehículos están en movimiento y, por lo tanto, el resultado de *DOT masks* y *All Masks* es muy similar. La fila inferior, en cambio, corresponde a una secuencia donde prácticamente todos los vehículos se encuentran aparcados. En este caso, *DOT masks* identifica adecuadamente que los automóviles

aunque potencialmente móviles se encuentran parados y actualiza sus máscaras para representarlos como objetos estáticos.

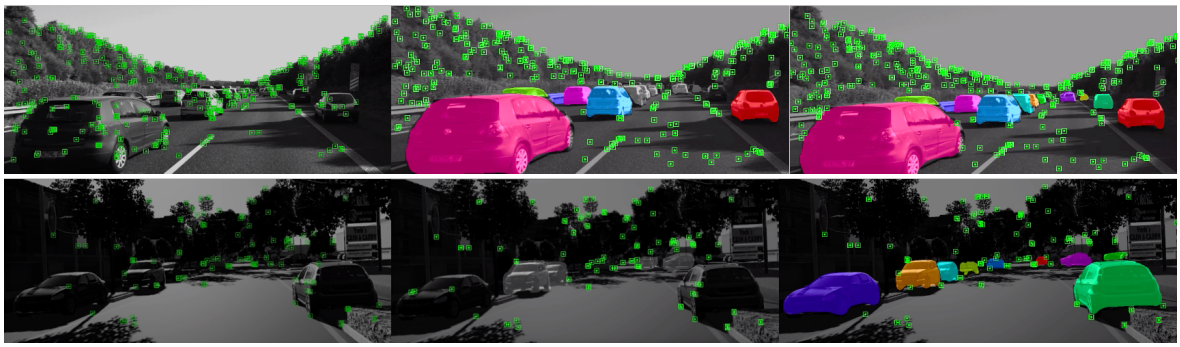

Figura 4.1: Ejemplos de las tres configuraciones utilizadas para la ejecución de ORB-SLAM2. Izq.: *No masks*. Centro: *DOT masks*. Dcha.: *All Masks*.

Evaluación. Como es habitual en los experimentos con sistemas en tiempo real de SLAM, para tener en cuenta su naturaleza no-determinista, se ha ejecutado 10 veces cada configuración en cada secuencia y como valor representativo se presenta la mediana de los resultados obtenidos. Todos los experimentos se han llevado a cabo en un ordenador portátil con procesador Intel Core i5 y 8GB de memoria RAM.

Métricas del error. La precisión de las trayectorias estimadas por ORB-SLAM2 respecto al *ground truth* con las tres configuraciones consideradas se evaluará utilizando dos métricas:

- **ATE** *(Absolute Trajectory Error)*: propuesto por [22] para valorar la consistencia global de la trayectoria, mide la distancia absoluta entre los puntos de la trayectoria real y la estimada previamente alineadas. Como medida representativa, los autores [22] calculan el valor cuadrático medio *(RMSE)* de los errores de posición de la cámara correspondientes a cada fotograma de la secuencia.

- **Errores relativos de rotación** t_{rot} $[°/100m]$ y **traslación** t_{trans} $[m/100m]$: miden la precisión local de la trayectoria en un intervalo fijo de distancia. Los autores [7] utilizan como valor representativo la media de los errores relativos calculados para subsecuencias de diferentes longitudes (100, 200, 400 y 800 metros) y tratan por separado los errores de rotación y de traslación.

Mientras que ATE suele utilizarse para evaluar sistemas de SLAM, los errores t_{rot} y t_{trans} resultan apropiados para cuantificar la deriva de los sistemas de odometría visual. Conviene hacer notar que aunque sean métricas diferentes, existe una fuerte correlación entre ellas. Los autores [22] afirman que no han encontrado diferencias

sustanciales entre el uso de una u otra y que, a menudo, al comparar la precisión de diferentes sistemas utilizando métricas distintas, el orden relativo no cambia. En los apartados 4.2.1 y 4.2.3 se ha optado por utilizar ATE como métrica mientras que en el apartado 4.2.2, para facilitar la comparación con los diversos estudios publicados, se incluyen ambos parámetros.

Para valorar la mejora que consigue DOT respecto de las otras dos configuraciones, se calcula la media de los errores (normalizados por el valor que obtiene DOT) para cada secuencia $\overline{\varepsilon}_{norm}$:

$$\overline{\varepsilon}_{norm} = \frac{1}{n} \sum_{i=0}^{n} \frac{\varepsilon_i}{\varepsilon_{DOT}} \tag{4.1}$$

4.2.1. Resultados con V-KITTI

Virtual KITTI [21][9] es un *dataset* sintético compuesto por 5 secuencias generadas a partir del *dataset* KITTI [7] para conducción autónoma. El cálculo de las trayectorias se ha llevado a cabo empleando la versión RGB-D de ORB-SLAM2, que recibe como entrada la imagen en color RGB y su correspondiente imagen de profundidades.

Los valores de ATE en la tabla 4.1 demuestran que, en promedio, los resultados de DOT son un 92.6 % y un 37.8 % mejores que los obtenidos con *No masks* y *All masks* respectivamente. Además, DOT obtiene el mejor resultado para 3 de las 5 secuencias evaluadas.

Seq.	ATE [m]			ATE/ATE_{opt}		
	No masks	DOT	All Masks	No masks	DOT	All Masks
01	**1.10**	1.14	1.38	1.00	1.04	1.26
02	0.16	0.14	**0.10**	1.60	1.43	1.00
06	0.11	**0.07**	0.08	1.67	1.00	1.18
18	4.77	**1.00**	1.50	4.79	1.00	1.51
20	29.42	**9.12**	13.54	3.23	1.00	1.49
$\overline{\varepsilon}_{norm}$	192.6 %	100.0 %	137.8 %			

Tabla 4.1: Comparación de los resultados de DOT en V-KITTI con los métodos de referencia: sin máscaras *(No masks)* y con todas las máscaras obtenidas de la red *(All Masks)*. Izq.: ATE [m]. Dcha.: ATE normalizado por el valor óptimo de cada secuencia.

Adaptación al contenido de la escena. La tabla 4.1 incluye también los valores de ATE normalizados con el valor más preciso en cada secuencia de entre las tres configuraciones. De esta forma, un valor igual a 1 indica el método con mejor resultado, mientras que desviaciones del valor unitario indican un peor desempeño. La escala de colores permite apreciar la posición relativa de los errores entre el mejor resultado (tonos

verdes) y el peor (tonos rojos). El dominio del color verde en la columna correspondiente a la implementación con DOT muestra la clara superioridad de esta opción respecto a las otras dos, evidenciando que si bien la utilización de máscaras puede resultar conveniente, la precisión mejora apreciablemente si se descartan únicamente los objetos que se ha comprobado que realmente están en movimiento. Estos resultados demuestran que la estrategia de DOT, por tanto, permite adaptar automáticamente el procesado a la opción óptima tanto para escenas estáticas como en casos con elementos móviles.

Debido a la gran variedad de casuísticas que producen los objetos dinámicos, se presenta a continuación un análisis detallado de las secuencias más relevantes.

Secuencias estáticas (secuencia 1): *escena urbana con numerosos coches aparcados a ambos lados de la calzada y con pocos coches en movimiento.* Atendiendo nuevamente a los resultados para la secuencia 1 en la tabla 4.1, la mayor precisión de ATE se obtiene con el método que no utiliza ninguna máscara (*No masks*, 1.10 m) y el peor para el que utiliza todas ellas (*All masks*, 1.38 m). El hecho de que ningún coche se mueva a lo largo de las secuencias las convierte en prácticamente estáticas y por lo tanto beneficia a la configuración *No masks*, que las considera como tal. Como se muestra en la figura 4.2, al poder extraer puntos de más áreas de la imagen, ORB-SLAM2 en esta configuración obtiene mayor precisión en la estimación del movimiento de la cámara. Ya que DOT comprueba que no hay objetos en movimiento, permite a ORB-SLAM2 utilizar los puntos localizados en los coches aparcados reduciendo así la pérdida de información útil y por lo tanto acercando la precisión a la solución óptima (*DOT*, 1.14 m).

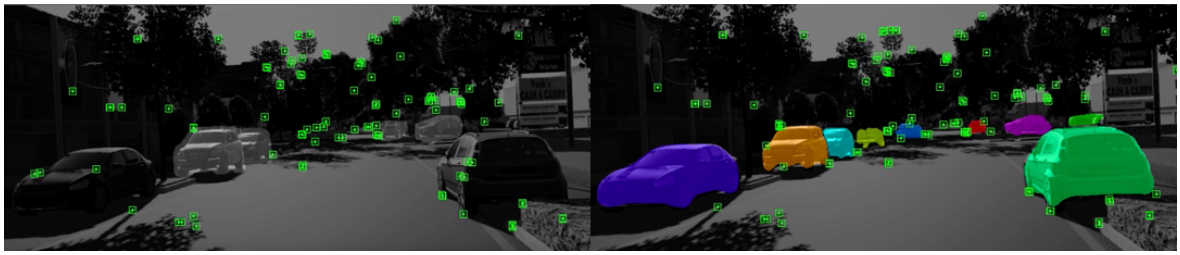

Figura 4.2: Fotogramas de la secuencia 1 del *dataset* V-KITTI en ORB-SLAM2. Izq.: *DOT masks*. Dcha.: *All Masks*.

Secuencias dinámicas (secuencias 6 y 18): *escenas de carretera extraurbanas donde todos los vehículos están en movimiento.* El hecho de que todos los coches de la escena se desplacen vulnera drásticamente la condición de estaticidad que necesita ORB-SLAM2 para calcular el movimiento de la cámara. Como se observa en la tabla 4.1, tanto usar todas las máscaras (*All masks*) como las generadas por DOT obtienen en este caso los mejores resultados. El elevado dinamismo de los objetos en la secuencia 18

provoca incluso el fallo de ORB-SLAM2 en la estimación del movimiento de la cámara en 6 de las 10 veces que se ha ejecutado (sólo se estima un 56 % de la trayectoria).

Secuencias parcialmente dinámicas (secuencias 2 y 20): *situación intermedia con coches parados y en movimiento. La secuencia 20 muestra un atasco en una autopista donde gran parte de la imagen está ocupada por vehículos en movimiento, pero donde también hay algunos que se encuentran localmente parados.* Mientras que no emplear ninguna máscara provoca un aumento del error al utilizar equivocadamente puntos localizados en objetos en movimiento, utilizar todas ellas sin verificar que el objeto se está moviendo provoca pérdida de información, especialmente cuando gran parte de la escena está compuesta por vehículos. Este tipo de situaciones demuestran claramente la versatilidad conseguida por DOT.

4.2.2. Resultados de KITTI *Odometry*

KITTI *Odometry* es un conjunto de secuencias de KITTI [7] especialmente diseñado para el desarrollo y evaluación de sistemas de odometría visual. En este caso, para la estimación de las trayectorias se ha ejecutado la versión estéreo de ORB-SLAM2, cuyas entradas son las imágenes registradas con un par estéreo de cámaras y una lista con sus registros temporales (*timestamps*).

La tabla 4.2 presenta los resultados de ATE para las 11 secuencias evaluadas en las diferentes configuraciones. Para facilitar la comparación, en la tabla 4.2 Dcha. se listan los valores de ATE de cada método normalizados por el ATE del método con mejor resultado.

Seq.	ATE [m]			ATE/ATE_{opt}		
	No masks	DOT	All Masks	No masks	DOT	All Masks
0	**1.77**	1.80	2.08	1.00	1.02	1.18
1	**6.37**	7.71	8.45	1.00	1.21	1.33
2	3.72	**3.70**	3.84	1.01	1.00	1.04
3	0.40	0.40	0.40	1.00	1.01	1.00
4	0.27	0.26	**0.24**	1.12	1.09	1.00
5	0.40	**0.39**	0.45	1.03	1.00	1.14
6	**0.63**	0.68	0.67	1.00	1.08	1.07
7	0.52	**0.51**	**0.51**	1.01	1.00	1.00
8	**3.04**	3.24	3.78	1.00	1.07	1.24
9	2.65	**0.98**	3.80	2.71	1.00	3.89
10	**1.23**	1.29	1.26	1.00	1.05	1.02
$\overline{\varepsilon}_{norm}$	112.7 %	100.0 %	130.3 %			

Tabla 4.2: Comparación de los resultados de DOT en KITTI *Odometry* con los métodos de referencia: sin máscaras *(No masks)* y con todas las máscaras obtenidas de la red *(All Masks)*. Izq.: ATE [m]. Dcha.: ATE normalizado por el valor óptimo para cada secuencia.

De acuerdo con los valores de ATE recogidos en la tabla 4.2, DOT obtiene un resultado global que es, respectivamente, un 12,7 % y un 30,3 % mejor que los métodos *No masks* y *All Masks*.

Los valores de t_{trans} y t_{rot}, recogidos en la tabla 4.3, reflejan el hecho de que este conjunto de secuencias, en comparación con V-KITTI, contienen muchos menos elementos dinámicos, por lo que el uso de las máscaras paradójicamente perjudica los resultados. De esta forma, DOT es capaz de igualar el nivel de resultados del método *No masks* puesto que ambos métodos obtienen un t_{trans} medio en torno al 0.76 m/100m y t_{rot} medio de 0.23 °/100m. El método *All masks* da lugar a errores superiores en ambos parámetros: en promedio, t_{trans} se sitúa en 0.81 m/100m y t_{rot} en 0.24 °/100m.

Seq.	No masks		DOT masks		All masks	
	t_{trans} [m/100m]	t_{rot} [°/100m]	t_{trans} [m/100m]	t_{rot} [°/100m]	t_{trans} [m/100m]	t_{rot} [°/100m]
0	**0.76**	0.23	0.77	**0.22**	0.84	0.24
1	**0.77**	**0.23**	0.78	**0.23**	0.85	0.24
2	**0.81**	0.23	0.82	0.23	0.86	0.23
3	0.82	0.23	0.82	0.23	0.82	0.23
4	**0.82**	0.23	**0.82**	0.23	0.86	0.23
5	**0.76**	**0.22**	**0.76**	**0.22**	0.80	0.23
6	**0.71**	0.22	**0.71**	0.22	0.74	0.22
7	0.71	0.22	**0.70**	0.22	0.73	0.22
8	0.74	0.23	**0.73**	0.23	0.77	0.23
9	0.76	**0.23**	**0.75**	**0.23**	0.79	0.25
10	0.76	**0.23**	**0.75**	**0.23**	0.80	0.25
$\overline{\varepsilon}_{norm}$	99.8 %	100.92 %	100.0 %	100.0 %	106.2 %	103.7 %

Tabla 4.3: Comparación del t_{trans} [m/100m] y t_{rot} [°/100m] de DOT en KITTI *Odometry* con los métodos de referencia: sin máscaras *(No masks)* y con todas las máscaras obtenidas de la red *(All Masks)*.

Precisión del GPS. De acuerdo con las especificaciones del *dataset*, los datos de posición recogidos por el GPS que utilizamos como referencia (*ground truth*) tienen una precisión de 10 cm. Por ello, se puede concluir que no existen diferencias significativas entre las tres configuraciones en las secuencias 3, 4, 5, 6, 7 y 10. Esto es el resultado, por una parte, del escaso número de objetos en movimiento y, por otra, de la rica textura de las imágenes, que proporciona una gran cantidad de puntos estáticos para estimar el movimiento de la cámara.

Cierre de bucle. De las 11 secuencias del *dataset*, hasta 7 presentan bucles a lo largo de su trayectoria. Cuando ORB-SLAM2 detecta que la cámara se encuentra en un punto en el que ya había estado previamente, se optimiza la trayectoria estimada hasta ese momento para hacerla compatible con la condición del cierre de bucle. Este

módulo de ORB-SLAM2 reduce la deriva ocasionada por una estimación deficiente y, por lo tanto, atenúa los efectos producidos por los objetos dinámicos o la eliminación de información útil de vehículos parados. Este hecho produce un aumento importante en la variabilidad de los errores debido a la enorme diferencia en la deriva de la trayectoria en función de si se detectan o no los bucles. La secuencia 9 es un ejemplo claro. Aunque es posible que la capacidad de DOT para lidiar con objetos dinámicos influya de forma significativa en el resultado, no se puede obviar el hecho de que, de las 10 ejecuciones, se cierra 6 veces el bucle en la configuración de DOT por tan sólo 4 de *No masks* y ninguna de *All masks* .

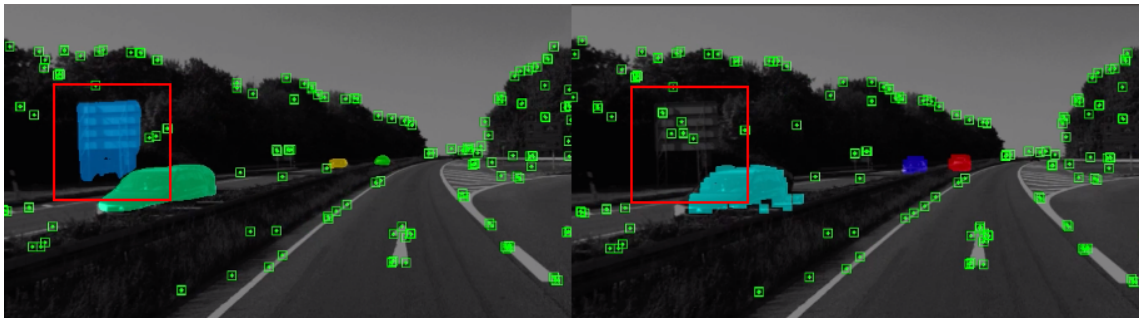

Figura 4.3: Comparación entre la configuración de *All Masks* y *DOT masks* donde se aprecia la robustez de DOT frente a errores de segmentación de la red.

Errores de la red neuronal (secuencia 01): *escena de carretera con todos los vehículos en movimiento.* Dado el elevado contenido dinámico de esta escena, cabría esperar que la mayor precisión se consiguiera con el método *All Masks*. Sin embargo, los resultados parecen estar intercambiados y es el método estático (*No masks*) el que alcanza la precisión óptima. Esta incoherencia se atribuye a que, en ocasiones, la red etiqueta erróneamente objetos estáticos útiles para estimar el movimiento de la cámara (por ejemplo, señales de tráfico o edificios) como objetos en movimiento. Como se muestra en la figura 4.3, DOT pemite corregir este efecto, etiquetando el objeto nuevamente como estático.

4.2.3. Resultados de KITTI *Raw*

El siguiente conjunto de secuencias se ha extraído de la sección *raw* del *dataset* KITTI [7] por su alto contenido de objetos móviles [23]. Al igual que para KITTI *Odometry*, se ha ejecutado la versión estéreo de ORB-SLAM2.

Como se muestra en la tabla 4.4, las diferencias entre secuencias y métodos son más evidentes al tratarse de un conjunto de secuencias especialmente seleccionadas por la abundancia de objetos en movimiento. Globalmente, DOT consigue una mejora de la precisión en ATE del 142,3 % respecto a *No mask* y del 15,9 % respecto a *All masks*.

Seq.	ATE [m]			ATE/ATE$_{opt}$		
	No masks	DOT	All Masks	No masks	DOT	All Masks
0926-0009	**1.23**	1.24	1.44	1.00	1.01	1.17
0926-0013	**0.26**	**0.26**	0.27	1.00	1.00	1.03
0926-0014	0.86	0.82	**0.78**	1.11	1.06	1.00
0926-0051	0.37	**0.36**	0.37	1.02	1.00	1.02
0926-0101	**8.66**	10.26	12.37	1.00	1.18	1.43
0929-0004	0.32	**0.30**	**0.30**	1.08	1.03	1.00
1003-0047	13.81	**1.25**	2.23	11.01	1.00	1.78
$\bar{\varepsilon}_{norm}$	242.3 %	100.0 %	115.9 %			

Tabla 4.4: Comparación de los resultados de DOT en KITTI *Raw* con los métodos de referencia: sin máscaras *(No masks)* y con todas las máscaras obtenidas de la red *(All Masks)*. Izq.: ATE [m]. Dcha.: ATE normalizado por el valor óptimo para cada secuencia.

Al igual que en los experimentos anteriores, la tabla 4.4 muestra cómo DOT se adapta al contenido de la secuencia obteniendo valores siempre cercanos o iguales al óptimo sin ser en ningún caso el peor.

Secuencias virtuales equivalentes. Las secuencias 0926-0009, 0929-0004 y 1003-0047 fueron utilizadas [21][9] para generar las secuencias sintéticas de V-KITTI (1, 18 y 20). Como cabría esperar, dado que el contenido de las escenas es idéntico, también lo es el análisis cualitativo de los resultados. En las secuencias 0926-0009 y 0929-0004, DOT interpreta correctamente el dinamismo de las escenas por lo que sus resultados se sitúan próximos al óptimo. El uso de máscaras en la secuencia 1003-0047 mejora drásticamente la precisión al ser la escena más dinámica y además DOT obtiene el mejor resultado al mantener la información de los coches parados en el atasco.

Capítulo 5

Contribuciones y trabajo futuro

El sistema DOT, cuyo desarrollo se ha iniciado en este trabajo, ha generado resultados que se consideran relevantes en su campo y que se tiene previsto presentar en una conferencia. También hay planes concretos para liberar el código, de forma que esté accesible para la comunidad científica. No obstante, se considera un trabajo todavía en desarrollo que, entre otros, se ampliará para buscar mejoras en precisión y robustez. Además de los mejoras de precisión conseguidas, que se han descrito en el capítulo anterior, existen otros aspectos de DOT que pueden considerarse contribuciones relevantes al estado de la técnica y que se resumen en esta sección, junto con algunas propuestas para la continuación del trabajo.

5.1. Implementación en tiempo real

La elevada carga computacional que supone el uso de redes neuronales impide su funcionamiento en tiempo real en plataformas de baja potencia (como teléfonos móviles o drones). De hecho, los trabajos actuales de SLAM en entornos dinámicos [2][5] no incluyen la segmentación semántica en el funcionamiento normal, sino que la ejecutan mediante un procesado previo de las imágenes.

Como se muestra en la figura 5.1 nuestra implementación permite que la segmentación se produzca cada cierto número de fotogramas, ya que DOT puede generar nuevas máscaras a partir de la estimación del movimiento de los objetos en la escena. Las máscaras proporcionadas por una red son muy exactas a baja frecuencia, cuando el tiempo de computación disponible es suficiente. Sin embargo, si se intenta optimizar el funcionamiento de la red puede empeorar significativamente la calidad de la segmentación. Por ello, creemos que la combinación de una red neural funcionando a baja frecuencia para producir segmentación de calidad junto con la estimación del movimiento de los objetos en los fotogramas intermedios puede ser una solución óptima.

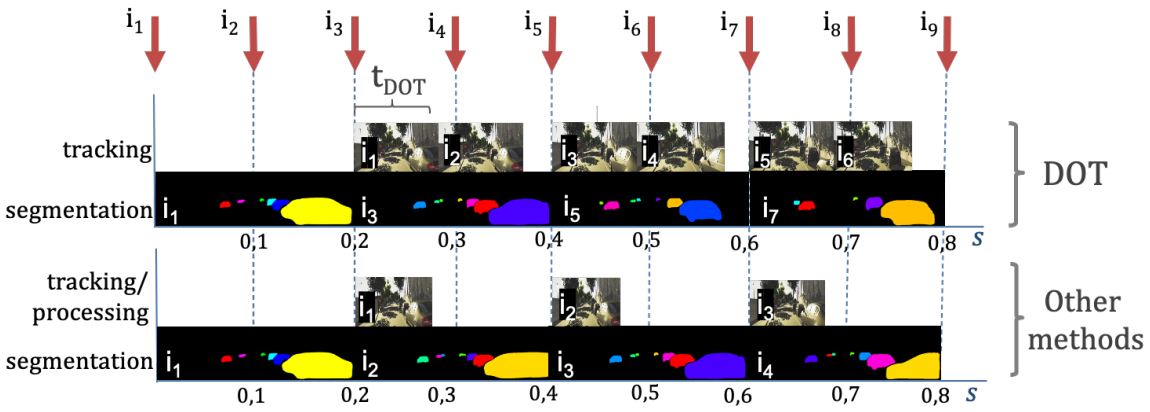

Figura 5.1: Secuencia temporal en la ejecución de DOT y su comparación con otros métodos.

5.2. Integración con odometría/SLAM visual

Un aspecto que podría ser conveniente revisar del algoritmo actual es el hecho de realizar por duplicado la estimación del movimiento de la cámara. Además de que DOT lleva a cabo inicialmente una estimación, ésta se repite en el sistema de SLAM enlazado.

Sin embargo, esta aparente falta de eficiencia aporta la ventaja de mantener la versatilidad para combinar DOT con cualquier sistema de odometría visual o SLAM del estado del arte. No obstante, para su implementación en aplicaciones concretas sin duda una integración total sería beneficiosa.

5.3. Evaluación intensiva

En este trabajo la evaluación se ha centrado en demostrar en qué medida "conocer el movimiento de los objetos" mejora la precisión de un sistema SLAM. No obstante, los buenos resultados obtenidos sugieren que sería interesante realizar evaluaciones adicionales para intentar comparar la combinación DOT/ORB-SLAM2 con otros sistemas de SLAM dinámico que componen el estado del arte actual [2][23][3][5].

Por otro lado aunque aquí se han utilizado *datasets* diseñados para conducción autónoma de vehículos [7][21], la formulación de DOT es perfectamente válida para otro tipo de aplicaciones con distintos tipos de objetos dinámicos, como la manipulación robótica en escenarios estructurados (por ejemplo, en oficinas) o la navegación de sistemas de vuelo de drones autónomos o teledirigidos. En ciertos casos, sería necesario volver a entrenar la red de segmentación, pero para muchos otros, la realización de experimentos con otras secuencias parece inmediata.

Finalmente, a pesar de haber alcanzado una buena precisión, la mejora en la estimación del movimiento de DOT podría perfeccionarse utilizando otras estrategias, como el uso combinado de *features* con los métodos directos o la implementación de un modelo cinemático que adapte las restricciones a la información extraída de la instanciación semántica (por ejemplo, si la red ha segmentado el objeto como "automóvil", se pueden constreñir los ángulos de alabeo o inclinación).

5.4. Métodos de segmentación semántica

DOT utiliza Mask R-CNN [6] como método de segmentación semántica. No obstante, existen otras redes neuronales que, aparte de la segmentación de objetos individuales, pueden aportar otro tipo de información, por ejemplo, acerca del movimiento de los objetos. Otra técnica posible consistiría en realimentar la red de segmentación con la información sobre el movimiento generada por DOT. Por ejemplo, si somos capaces de estimar la posición del objeto en el siguiente fotograma, la red podría emplear esta información para buscar el objeto en esta región, optimizando así su funcionamiento.

Capítulo 6

Conclusiones

En "SLAM visual robusto en entornos dinámicos combinando deep learning y consistencia multivista" se ha desarrollado "Dynamic Object Tracking (DOT)", un sistema de *front-end* que combina instancias semánticas con geometría multivista para estimar el movimiento de una cámara y los objetos de la escena utilizando métodos directos.

La evaluación de DOT en combinación con ORB-SLAM2 [1] en tres datasets públicos diseñados para la investigación en conducción autónoma [7][21][9] muestra cómo la información generada por DOT acerca del movimiento de los objetos permite adaptar el funcionamiento del SLAM al contenido de la escena y mejorar sensiblemente tanto precisión como robustez.

La independencia de DOT con respecto del SLAM lo convierte en un *front-end* versátil que se puede adaptar con un trabajo de integración mínimo a cualquier odometría visual o sistema de SLAM del estado del arte. Además, DOT permite que la segmentación semántica (habitualmente con un elevado coste computacional) pueda realizarse a una frecuencia menor que la de la cámara, lo que a diferencia de otros sistemas facilita su ejecución en tiempo real.

DOT corrige errores de la segmentación semántica que afectan al SLAM, al considerar estáticos objetos que habían sido mal etiquetados por la red, por ejemplo edificios o señales instanciados como automóviles.

Finalmente, creemos que la extensión y mejora de las capacidades de DOT puede aportar beneficios apreciables no solamente para el uso en conducción autónoma sino también en aplicaciones de robótica o realidad aumentada.

Bibliografía

[1] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[2] Berta Bescós, José M. Fácil, Javier Civera, and José Neira. DynSLAM: Tracking, Mapping and Inpainting in Dynamic Scenes. *CoRR*, abs/1806.05620, 2018.

[3] Martin Rünz, Maud Buffier, and Lourdes Agapito. MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects, 2018.

[4] NavVis GmbH. URL: https://www.navvis.com [Online. Accedido el 13/06/2020.].

[5] Binbin Xu, Wenbin Li, Dimos Tzoumanikas, Michael Bloesch, Andrew Davison, and Stefan Leutenegger. MID-Fusion: Octree-based Object-Level Multi-Instance Dynamic SLAM, 2018.

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[7] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? the KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[8] Preetham Govind K.S. Semantic Segmentation Techniques For Computer Vision Tasks. URL: https://link.medium.com/B1wmFsM4C7 [Online. Accedido el 15/05/2020.].

[9] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual KITTI 2, 2020.

[10] Raluca Scona, Mariano Jaimez, Yvan R. Petillot, Maurice Fallon, and Daniel Cremers. StaticFusion: Background Reconstruction for Dense RGB-D SLAM in Dynamic Environments. In *2018 ICRA*. IEEE.

[11] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. ElasticFusion. *Int. J. Rob. Res.*, 35(14):1697–1716, 2016.

[12] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, 2013.

[13] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[14] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *CoRR*, 2015.

[15] Inc. Matterport. Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow, 2019. URL: https://github.com/matterport/Mask_RCNN [Online. Accedido el 03/12/2019].

[16] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context, 2014.

[17] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[18] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct Sparse Odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.

[19] Alejo Concha and Javier Civera. An evaluation of robust cost functions for RGB direct mapping. In *2015 European Conference on Mobile Robots (ECMR)*, pages 1–8. IEEE, 2015.

[20] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[21] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2016.

[22] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, 2012.

[23] Jiahui Huang, Sheng Yang, Zishuo Zhao, Yu-Kun Lai, and Shi-Min Hu. ClusterSLAM: A SLAM Backend for Simultaneous Rigid Body Clustering and Motion Estimation. 2019.